Winter 2019 - CSE 641 Deep Learning
End Sem - Apr. 25, 2019

Maximum score: 105 **Time: 120 mins**

Instructions:

- Try to attempt all questions.
- Keep your answers short, succinct and to the point. Unnecessarily long and convoluted answers may cost you some points. Make sure that you define all variables in your equations.
- Do not copy. Institute plagiarism policy is strictly enforced.
- In the unusual case that a question is not clear, even after discussing with the invigilators, please state your assumptions *clearly* and solve the question. Reasonable assumptions will be accounted for while grading.

1. (*30 points*) **Deep Metric Learning and Domain Adaptation**:

a) (*5 points*) What is the distance metric learning problem? Write the mathematical expression for the learned distance metric.

Answer *Distance metric learning problem* (2.5)

Distance metric learning is a problem of learning distances given a dataset. Given a dataset $X = \{x_1, ..., x_N\}$, certain pairwise constraints are collected in the sets of similar and dissimilar pairs respectively.

$$\begin{aligned} S = (x_i, x_j) : &\quad x_i \text{ and } x_j \text{ are similar,} \\ D = (x_i, x_j) : &\quad x_i \text{ and } x_j \text{ are not similar.} \end{aligned}$$

Mathematical expression for the learned distance metric (2.5)

Mahalanobis Distance (Linear Metric Learning):

$$d_M(\boldsymbol{x}, \mathbf{x}') = \sqrt{(\mathbf{x} - \mathbf{x}')^T \mathbf{M} (\mathbf{x} - \mathbf{x}')}$$

where $\mathbf{M}$ is symmetric PSD matrix.

Nonlinear Metric Learning: Kernelization

$$d_M(\phi_i, \phi_j) = \sqrt{(\mathbf{k_i} - \mathbf{k_j})^{\mathbf{T}} \mathbf{M} (\mathbf{k_i} - \mathbf{k_j})}$$

where $\mathbf{k}_i = \phi^T \phi_i = [K(x_1, x_i), ...K(x_n, x_i)]^T$ and $K(x, x') =< \phi(x), \phi(x') >$ is a kernel.

b) (*5 points*) Give a loss function that will help you train a Deep Neural Network that learns a distance metric.

Answer *Loss Function* (2.5)

The loss function can be defined as:

$$L(x_i, x_j, y) = y||f(x_i) - f(x_j)||_2^2 + (1 - y)max(0, (k - ||f(x_i) - f(x_j)||_2)^2$$

where x_i and x_j are two data samples, $||f(x_i) - f(x_j)||_2^2$ is the squared Euclidean distance, y is the label indicating whether they are similar or dissimilar, f is the feature space defined by the neural network, and k is the margin.

Explanation (2.5):
Define:
$y = 1$ if x_i and x_j are similar and
$y = 0$ if x_i and x_j are dissimilar
Case 1: x_i and x_j are similar and $y = 1$
Only the first term of the loss function will be active. f will be learned to minimize the distance between x_i and x_j
Case 2: x_i and x_j are dissimilar and $y = 0$ Only the second term of the loss function will be active. Now to minimize the loss, $||f(x_i) - f(x_j)||_2^2$ should be maximized as k is a constant. So, f will be learned to maximize the distance between x_i and x_j.
One could also use the contrastive loss here.

c) (*5 points*) What is the Siamese architecture? Why is it useful for distance metric learning?
Answer *Siamese architecture (1.5)*

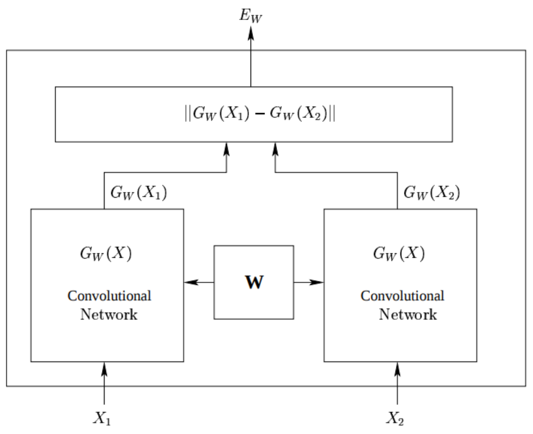


Figure 1: Siamese architecture

Explanation of architecture with salient points (1)
It consists of two identical convolutional networks with tied weights and bias which means that both networks are symmetric, hence the name Siamese architecture. A contrastive loss function based on E_W is used for training this architecture such that energy of similar pairs decreases and the energy of dissimilar pairs increases.

Usefulness for distance metric learning (1.5)
Let W be the shared parameter vector that is subject to learning, and let $G_W(X_1)$ and $G_W(X_2)$ be the two points in the low-dimensional space that are generated by mapping X_1 and X_2. Then the architecture can be viewed as a scalar energy function $E_W(x_1, x_2)$ that measures the compatibility between X_1 and X_2. Minimization of the loss function defined as using E_W as:

$$L(W, x_1, x_2, y) = y\lambda_1(E_W)^2 + (1-y)\lambda_2 \exp(-\lambda_3 E_w)$$

Where y is 1 for a similar pair and 0 for dissimilar pair, will decrease the energy of similar pairs and increase the energy of dissimilar pairs.

Importance of shared weights/symmetry/siamese architecture (1)
Symmetry is important as it makes the network invariant to switching the input images. Moreover, this characteristic makes the networks much faster to train since the number of parameters is reduced significantly.

d) (*5 points*) What is meant by domain shift? How does domain adaptation help in dealing with problems of domain shift?

Answer *Domain Shift: (2.5)*
Domain shift refers to the existence of a difference between distributions of data from two different domains. For example, when a model is trained on images taken from cars' dashcam and tested on images taken from a surveillance camera mounted on the roadside, the distributions of the data might be quite different, despite similarities in factors like image resolution, focal length, camera height from ground, etc. This may lead to a drop in performance when the model is tested on unseen data from a different domain (surveillance camera videos here).

Domain adaptation (2.5)
Domain adaptation is a technique that attempts to align feature space distributions across the source and target domains. Let D_s denote a source distribution and D_T a target distribution. A representation function R is a function which maps instances to features $R : X \to Z$. Let D_s and D_T be the induced distribution over the feature space Z for source distribution and target distribution respectively. Let $f : X \to [0, 1]$ be the labeling rule, common to both domains, and $\tilde{f}$ is the induced image of f under the representation R. A predictor is a function, h, from the feature space, Z to $[0, 1]$. We denote the probability, according to the distribution D_s, that a predictor h disagrees with f by:

$$\begin{aligned}\epsilon_s(h) &= E_{z\sim\tilde{D}_s}[E_{y\sim\tilde{f}_z}[y \neq h(z)]] \\ &= E_{z\sim\tilde{D}_s}|\tilde{f}(z) - h(z)|\end{aligned}$$

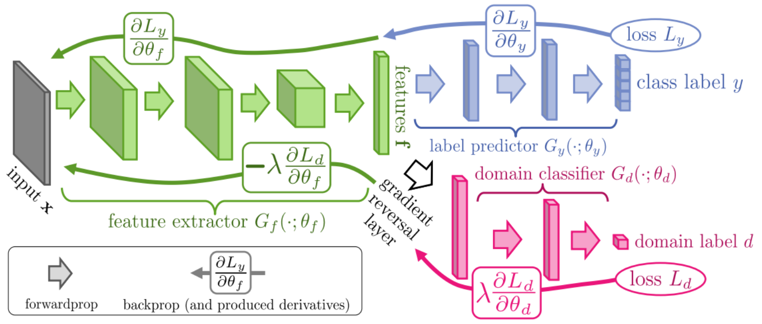
Similarly define $\epsilon_T(h)$. The aim of domain adaptation is to find a single hypothesis $h \in H$ which performs well on both domains i.e.

$$inf\{h \in H\}[\epsilon_s(h) + \epsilon_T(h)] \leq \lambda$$

e) (*10 points*) Design a neural network that learns domain invariant features for a classification task by applying adversarial training. What is the loss function? Write the gradient based update equations for the different parts of the network. Feel free to group together parameters into logical blocks.

Answer *Design a neural network that learns domain invariant features for a classification task by applying adversarial training. (2.5)*

Explanation (2.5)
The architecture includes a deep feature extractor (green) and a deep label predictor (blue), which together form a standard feed-forward architecture. Unsupervised domain adaptation is achieved by adding a domain classifier (red) connected to the feature extractor via a gradient reversal layer that multiplies the gradient by a certain negative constant during the backpropagation-based training. Otherwise, the training proceeds standardly and minimizes the label prediction loss (for source examples) and the domain classification loss (for all samples). Gradient reversal ensures that the feature distributions over the two domains are made similar (as indistinguishable as possible for the domain classifier), thus resulting in the domain-invariant features.



$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features $\mathbf{f}$
label predictor $G_y(\cdot;\theta_y)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
domain classifier $G_d(\cdot;\theta_d)$
input $\mathbf{x}$
gradient reversal layer
feature extractor $G_f(\cdot;\theta_f)$
domain label d
$\lambda \frac{\partial L_d}{\partial \theta_d}$
loss L_d
forwardprop
backprop (and produced derivatives)

Figure 2: ADNN Architecture

Loss function (2.5)
Let $G_f(\cdot, \theta_f)$ be the D-dimensional neural network feature extractor, with parameters θ_f. Also, let $G_y(\cdot, \theta_y)$ be the part of DANN that computes the networks label prediction output layer, with parameters θ_y, while $G_d(\cdot, \theta_d)$ corresponds to the computation of the domain prediction output of the network, with parameters θ_d.
The prediction loss and the domain loss respectively are:

$$L_y^i(\theta_f, \theta_y) = L_y(G_y(G_f(\mathbf{x}_i; \theta_f); \theta_y), y_i),$$
$$L_d^i(\theta_f, \theta_d) = L_d(G_d(G_f(\mathbf{x}_i; \theta_f); \theta_d), d_i).$$

The DANN can then be trained by minimizing the following loss function:

$$E(\theta_f, \theta_y, \theta_d) = \frac{1}{n}\sum_{i=1}^{n} L_y^i(\theta_f, \theta_y) - \lambda(\frac{1}{n}\sum_{i=1}^{n} L_d^i(\theta_f, \theta_d) + \frac{1}{n}\sum_{i=n+1}^{N} L_d^i(\theta_f, \theta_d)),$$

by finding the saddle point $\tilde{\theta}_f$,$\tilde{\theta}_y$,$\tilde{\theta}_d$ such that

$$(\tilde{\theta}_f, \tilde{\theta}_y) = \arg\min\{\theta_f, \theta_y\} E(\theta_f, \theta_y, \tilde{\theta}_d)$$
$$\tilde{\theta}_d = \arg\min\{\theta_d\} E(\tilde{\theta}_f, \tilde{\theta}_y, \theta_d)$$

Gradient-based update equations for the different parts of the network (2.5)

$$\theta_f \leftarrow \theta_f \quad - \quad \mu \left(\frac{\delta L_y^i}{\theta_f} - \lambda \frac{\delta L_d^i}{\theta_f} \right)$$
$$\theta_y \leftarrow \theta_y \quad - \quad \mu \left(\frac{\delta L_y^i}{\theta_f} \right)$$
$$\theta_d \leftarrow \theta_d \quad - \quad \mu\lambda \left(\frac{\delta L_d^i}{\theta_d} \right)$$

where μ is the learning rate.

2. (*25 points*) **GANs and Disentangled Representations**

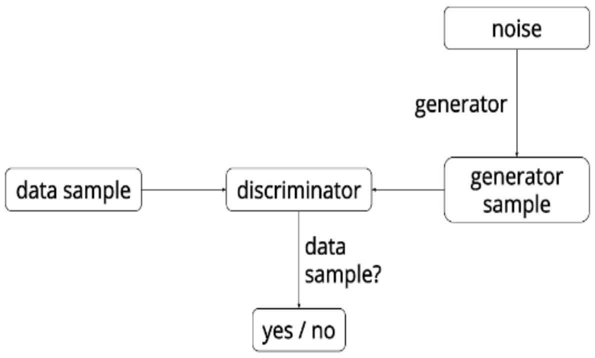


Figure 3: Block diagram for working of GAN

a) (*5 points*) Explain the working of a Generative Adversarial Network (GAN) using a block diagram. Write the expression of the loss function and its explanation.

Answer: The generator will try to generate fake images that fool the discriminator into thinking that they are real. And the discriminator will try to distinguish between a real and a generated image as best as it could when an image is fed. They both get stronger together until the discriminator cannot distinguish between the real and the generated images anymore.

The generator will try to generate fake images that fool the discriminator into thinking that theyre real. And the discriminator will try to distinguish between a real and a generated image as best as it could when an image is fed. They both get stronger together until the discriminator cannot distinguish between the real and the generated images anymore.

GAN loss:

$$\min_{\theta_g} \max_{\theta_d} \quad [E_{x \sim p_{data}} \log D_{\theta_d}(x) + E_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))] \tag{1}$$

$D_{\theta_d}(x)$ Discriminator output for real data x

$D_{\theta_d}(G_{\theta_g}(z))$ Discriminator output for generator fake data

The discriminator tries to maximize the objective function so that the gradient ascent can be performed on the objective function and generator tries to minimize the objective function so that gradient descent can be performed. By alternating the gradient ascent and descent the network is trained.

b) (*2 points*) Do GANs disentangle different attributes automatically? Argue why or why not?

Answer: In general, no. There is no mechanism in the training of general GAN framework that constraints the latent space to be disentangled. However, with additional constraints, e.g., the ones in InfoGAN can learn latent spaces that are disentangled.

c) (*3 points*) List three tricks/hacks to stabilize training of GANs.

Answer: Any three mentioned from this repo: https://github.com/soumith/ganhacks

d) (*10 points*) Describe a technique for disentangling factors of variation. **Answer**: Any technique will work, e.g., Matthieu's paper on VAE+adversarial training for disentangling; Jha's paper on VAE+cycle consistency for disentangling.

3. (*20 points*) Autoencoder (AE) and Variational AE (VAE)

a) (*2+1+2=5 points*) What kind of a loss function is used to train Autoencoders? Why is it necessary to constraint the latent space of an AE? Give two popular ways of constraining a regular autoencoder.

Answer: Typically, an MSE loss is used for training AEs.

If the latent space is not constrained, then the AE runs the risk of learning an identity function.

Low dimensionality and sparsity constraints. Set the layer between the encoder and decoder of a dimension lower than the inputs one. Then trash the decoder and use that middle layer as output layer.

b) (*1+1+3=5 points*) What is a VAE? Why is the KL divergence term required for the VAE? Explain why is the reparametrization trick needed. Give necessary equations used for implementing the reparametrization while training VAEs.

Answer: VAE is a generative model that uses a neural network to perform variational approximation of the posterior distribution. KL divergence is used as a measure between the approximated posterior distribution $q_\phi(z|x)$ and the true posterior $p(z|x)$.

VAE has an encoder and decoder architecture, which requires sampling from the approximate posterior estimated by the encoder. To implement encoder and decoder as a neural network, one needs to backpropagate through the entire architecture, however, sampling is a non-differentiable operation. In order to avoid this hurdle, we use reparameterization trick.

The reparametrization trick involves a linear transformation of samples from a standard normal distribution, followed by addition of the mean.

$z{\sim}\mathcal{N}(\mu,\Sigma) \Rightarrow z = \mu + L\epsilon$, where, $\epsilon \sim \mathcal{N}(0, I)$, and the covariance $\Sigma = LL^\top$

Now instead of saying that Z is sampled from q(z|ϕ,x), we can say Z is a function that takes a parameter (ϵ,(μ, L)) and this μ, L comes from an upper neural network (encoder). Therefore while backpropagation all we need is partial derivatives w.r.t. μ, L and ϵ are irrelevant for taking derivatives.

c) (*10 points*) My desired latent space distribution is given by $p(z) = 0.9\mathcal{N}(0, \mathbf{I}) + 0.1\mathcal{U}(-6, 6)$, where $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ and $\mathcal{U}(a, b)$ is a multivariate uniform distribution with non-zero density in the range $[a, b]$ for each dimension. This type of a mixture distribution is often used in robust statistics to model outliers. Can I design a VAE with $p(z)$ as my latent space distribution? For this modified VAE problem, provide details of: Output of the Encoder, KL-divergence term in the loss function, Reparametrization trick, the training process and the test time use. {*Hint*: $KL(p||q) = -\sum_{x\in\mathcal{X}} p(x) \log\left(\frac{q(x)}{p(x)}\right)$, where $p(x)$ and $q(x)$ are discrete probability mass functions. Similarly, the KL-divergence can be defined for continuous variables and PDFs.}

Answer: Yes we can design the VAE given p(z) as our latent space distribution is a PDF. Encoders output: μ(x), $\sum$(x), $a(x)$, $b(x)$.

KL divergence term in the loss function is: KL((q$_\theta$(z|x$_i$))||$p(z)$), here p(z) can be replaced by the given latent space distribution p(z).

For the reparametrization:

For gaussian $\epsilon{\sim}\mathcal{N}$(0,I), z$_1$ = $\epsilon\sigma_x$+μ_x

For uniform $z_2 = (b - a) * \nu + a$, whre $\nu \sim \mathcal{U}(0, 1)$

z = 0.9z_1 + 0.1z_2

While training the input x passed through the encoder and the outputs from the encoder will help to estimate z by reparametrization trick and then z is pushed through the decoder to compute reconstruction and variational loss.

The test sample out of the given distribution p(z) and pass it to the decoder.

4. (*30 points*) **Design**

Your goal is to track an object in a dashcam (a camera installed on the dashboard of a vehicle) video. Object tracking requires drawing a tight bounding box around the object in each frame of a video. Using your knowledge of deep learning, design an end-to-end model for tracking an object in the video feed. You would need to present:

- (*10 points*) the network architecture you choose, describing each of the components (different types of layers, but not necessarily the number of layers) and their need
- (*4 points*) the loss function that you would use
- (*2 points*) the type of annotation that you would require
- (*4 points*) a list of engineering tricks, data augmentation, and training strategies that you would consider
- *(10 points)* Suppose you want to perform unsupervised or semi-supervised domain adaptation on this model so that an object tracking model trained on roads in the US can work in the Indian context. How would you design a domain adaptation model for this problem?

{*Hint*: You may assume that the video has a lot of redundancy, as the object speeds are relatively low compared to the frame rate of the camera. For the domain adaptation, you don't necessarily need to stick to the adversarial approach.}

Answer: This is an open-ended question. Following is the rubric:

- Since object tracking is the task, an object (class label notwithstanding) should be tracked, i.e., bounding box should be determined in each frame for *the same* object instance. For this, both temporal and visual context has to be taken into account. If you have used only one of the two, you would have received about half the points. Depending on the level of detail, you may have gotten $5 \pm \delta$.
- The loss function should include localization and *object instance* identification (not necessarily category), which can be obtained by a temporal term.
- You would need location of the object at each frame, as well as the object instance ID.
- Simply applying the classfication domain adaptation of Q 1. e) does not suffice. It applies domain invariance to the entire image. You need to account for the object proposals, i.e., adapting features at the region level.